OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(12) (19) (CA) **Demande-Application**

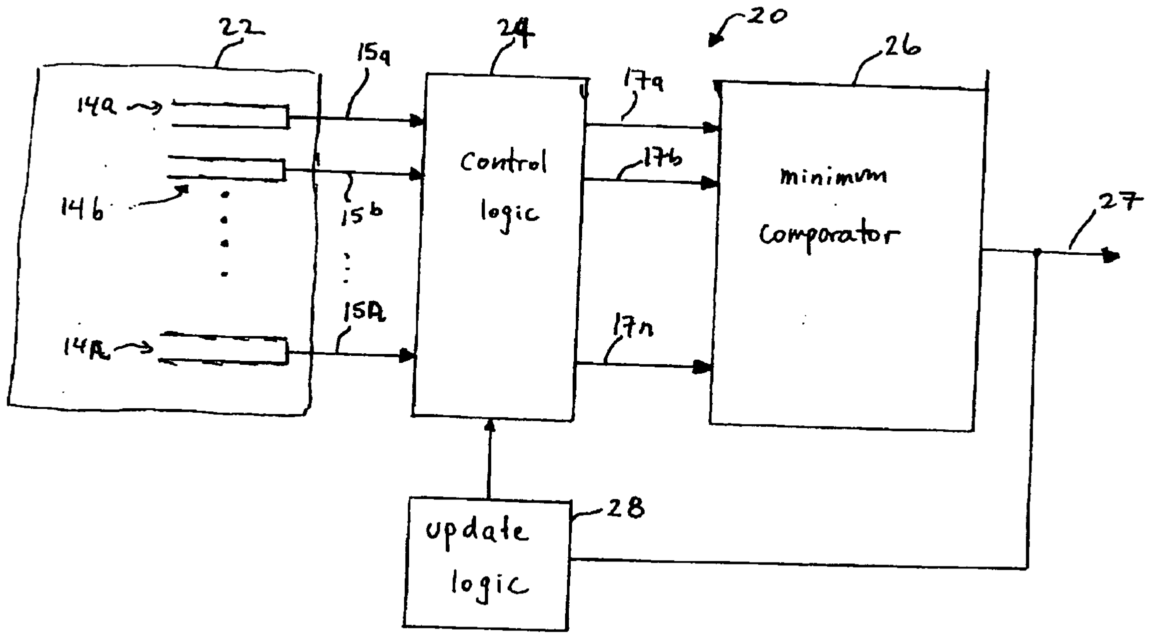
(21) (A1) **2,302,129**
(22) 2000/03/27
(43) 2000/09/26

(72) JANOSKA, MARK, CA
(72) CHOW, HENRY, CA
(72) HUNG, ANTHONY, US
(71) NEWBRIDGE NETWORKS CORPORATION, CA

(51) Int.Cl.^{7} H04L 12/24, H04L 12/56
(30) 1999/03/26 (2,267,021) CA

(54) ARBITRAGE D'HORODATAGE POUR DISPOSITIFS DE COMMUNICATIONS NUMERIQUES

(54) TIMESTAMPING ARBITRATION IN DIGITAL COMMUNICATION DEVICES



(57) A hardware structure for an n-level hierarchical scheduler is shown. The hardware uses a time-stamping arbitration mechanism for servicing a plurality of queues. Work-conserving and non-working conserving embodiments of the scheduler are shown.

ABSTRACT

A hardware structure for an n-level hierarchical scheduler is shown. The hardware uses a time-stamping arbitration mechanism for servicing a plurality of queues. Work-conserving and non-working conserving embodiments of the scheduler are shown.

TIMESTAMPING ARBITRATION IN DIGITAL COMMUNICATION DEVICES

FIELD OF INVENTION

The invention generally relates to digital communication packet scheduling systems, and more particularly to hierarchical schedulers which employ timestamping techniques for arbitrating amongst queues contending for service.

BACKGROUND OF INVENTION

Hierarchical schedulers, which comprise a plurality of schedulers interconnected in a multi-tiered arrangement, find use in variety of packet switching devices. See, for instance, Floyd and Jacobson, "*Link Sharing and Resource Management Models for Packet Networks*", I.E.E.E./ACM transactions on networking, vol. 3, no. 4, August 1995; and Bennett and Zhang, "*Hierarchical Packet Fair Queuing Algorithms*", proceedings of ACM SIG COMM, pages 143 to 156, Stanford, CA, August 1996.

It is relatively easy to implement schedulers in software using conventional programming languages such as 'C' executing on a general purpose data processor. However, the software approach is inherently limited in terms of speed of operation. Thus, in a high speed packet switching device it is preferred to implement a scheduler using dedicated hardware.

One particular technique conducive to the practical implementation of a scheduler or arbiter is timestamping arbitration. In this technique, each data packet is associated with a theoretical emission time (TET) which essentially specifies a future time that the packet is destined for service. A scheduler may then sort the TETs in order to determine which packet, and hence which queue, should be serviced at the next available scheduling time slot. The timestamping technique can be applied to non-work conserving schedulers, such as shapers, or work conserving schedulers, such as weighted fair queuing schedulers. The prior art has not, however, presented an effective solution to providing a high speed hierarchical scheduler which employs the time-stamping technique.

It is thus desired to provide a general purpose method for implementing timestamping arbitration using a hardware structure which can be applied to both shapers and weighted fair queuing (WFQ) schedulers. Furthermore, the hardware structure should also be capable of implementing shaping and WFQ scheduling in a hierarchical manner.

SUMMARY OF INVENTION

Broadly speaking, the invention provides an n-level hierarchical scheduler for servicing a plurality of queues. The scheduler includes control logic for assembling a tag for each queue, said tag having a field indicative of the eligibility of the queue and n fields, one for each level of the hierarchical scheduler, indicative of the priority of the queue with respect to the corresponding level of the hierarchy. A comparator evaluates the tags and provides an output specifying a winning queue. Update logic receives the comparator output and updates the priority fields of the tag.

In the preferred embodiment, the comparator is a minimum comparator and the fields are concatenated such that the eligibility field assumes a more significant position than the n priority fields in the tag, and the significance of the n priority fields corresponds to their respective level in the hierarchy. The priority field representing the bottom-most level for a given queue is set to a theoretical emission time of the given queue. The theoretical emission time may be computed in relation to a shaper scheduler or a weighted fair queuing scheduler.

BRIEF DESCRIPTION OF DRAWINGS

The foregoing and other aspects of the invention will become more apparent with reference to the detailed description of its preferred embodiment in conjunctions with the drawings, in which:

Fig. 1 is a diagram of the logical structure of a hierarchical scheduler;
Fig. 2 is a schematic block diagram of a hardware structure according to the preferred embodiment for implementing the hierarchical scheduler shown in Fig. 1;
Fig. 3 is a diagram of a data structure or tag which is associated with each queue and evaluated by the hardware structure shown in Fig. 2;
Fig. 4 is a functional block diagram of the hardware structure shown in Fig. 2;
Fig. 5A is a schematic diagram of a non-work conserving hierarchical scheduler;
Fig. 5B is a diagram of records associated with each queue and with certain sub-schedulers of the hierarchical scheduler shown in Fig. 5A;
Fig. 6A is a schematic diagram of a work conserving hierarchical scheduler;
Fig. 6B is a schematic diagram of records associated with each queue and with certain sub-schedulers of the hierarchical scheduler shown in Fig. 6A; and

Fig. 7 is a series of time lines showing the data processing provided by the hardware structure as applied to implement the work conserving hierarchical scheduler shown in Fig. 6A.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Fig. 1 shows a logical, generic, model of a hierarchical scheduler 10 which comprises a plurality of schedulers 12 (each of which is termed herein as a "sub-scheduler") interconnected in a multitiered arrangement. Each sub-scheduler 12 is a logically independent scheduler and serves one or more entities directly beneath it in the hierarchy. In the illustrated embodiment a two-level hierarchy is shown. The sub-schedulers 12 on the bottom-most level of the hierarchy, i.e., level 2, serve queues 14 (individually labeled 14a, 14b, ... 14n) which hold data packets 16 that are preferably, although not necessarily, equally sized. The sub-scheduler 12 on the higher level, i.e., level 1, serves the sub-schedulers 12 situated on the immediately lower level. In practice, the sub-schedulers 12 preferably only pass the identity of a queue (i.e., "queue identifier) to be serviced upstream to the top-level sub-scheduler and that entity, or alternatively an external entity, is responsible for moving the head-of-line (HOL) data packet from a queue 14 selected for service (i.e., a "winning" queue) to an output stream 18.

Fig. 2 is a schematic block diagram of a structure 20 according to the preferred embodiment for implementing the hierarchical scheduler 10 in hardware. As will be explained in greater detail below, the hardware structure 20 may be employed in relation to a work-conserving hierarchical scheduler wherein each sub-scheduler 12 thereof is, for example, a weighted fair queue (WFQ) scheduler, or a non-work conserving hierarchical scheduler wherein sub-schedulers 12 include, for example, shaper schedulers. In either case, the hierarchical scheduler 10 employs a time-stamping technique for arbitrating amongst contending queues 14. Time-stamping techniques for non-hierarchical shaper and WFQ schedulers are respectively disclosed, for instance, in Stiliadios, D. and Varma, A., "*A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms*", Proceedings of I.E.E.E. INFOCOM, Japan, 1997, and Goyal et al., "*Start-Time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks*", IEEE/ACM Trans. Networking, Vol. 5, No. 5, October 1970, which disclosures are incorporated herein by reference. It will be appreciated that in these time-stamping techniques each packet is associated with a time-stamp or theoretical emission time (hereinafter TET) which is used by a scheduler for the arbitration decision. In the preferred embodiment, however, time stamping is performed per queue rather than per packet. Generally speaking, this means that each queue is associated with a single TET which is equivalent to the time-stamp that the prior art would have granted to the head-of-line (HOL) packet in the queue. The inventors have found that time stamping per queue is likely more economical to implement in practice because of lower memory storage requirements. The TETs associated with queues 14 are represented in Fig. 2 by reference no. 15 (individual TETs being labelled 15a, 15b, ... 15n).

The hardware structure 20 comprises a control logic block 24 which uses the TETs 15 associated with each queue 14 to build a data structure or tag 17 for each queue 14 (individual tags being labeled 17a, 17b, ... 17n). Generally speaking, each tag 17 represents the priority of the corresponding queue 14 with respect to the multiple levels of the scheduling hierarchy. For example, the structure of the tag 17 for a two level hierarchical scheduler is shown in Fig. 3 and comprises an eligibility field (ELN) 36, a level 1 TET field 37, and a level 2 TET field 38. More generally, in an n-level hierarchical scheduler the tag 17 would include n TET fields, one for each level of the hierarchy. The control logic block 24 sets the eligibility field 36 to indicate whether a given queue 14 is eligible to be serviced; for instance, if it has at least one data packet queued therein. The level 2 or bottom-most level TET 38 is set to the actual TET 15 associated with the corresponding queue. The level 1 TET field 37 is associated with the level 1 sub-scheduler 12 and its computation will be described in greater detail below. The tags 17, which can be considered to provide effective TET values, are evaluated by a minimum comparator 26 (Fig. 2) which selects the winning queue. The ordering or position of the fields is important in the tag 17 because the concatenation of its fields provides a value for a given queue 14 which can be quickly evaluated by the minimum comparator 26 relative to the tags 17 of other queues. This enables queue arbitration to be quickly preformed which can be advantageous for systems that employ hundreds or even thousands of queues, as may occur, for instance, in a high speed ATM network node or switch.

In the preferred embodiment, the minimum comparator 26 provides an identifier of the winning queue on its output line 27. The queue identifier is read by an update logic block 28 which is responsible for updating the level 1 and level 2 TETs 37 and 38, as described in greater detail below.

In practice, the control logic block 24 preferably communicates with a control memory 22 which stores pertinent information concerning each queue 14, such as its queue depth counter, and which level 2 sub-scheduler 12 is responsible for servicing the queue. The particular information stored in the control memory will depend on the type of hierarchical shaper being implemented, as described below. The preferred implementation of the hardware structure 20 is shown in greater detail in the functional block diagram of Fig. 4. In this case, the data packets 16 are temporarily stored in a common data memory 30 and queue memory manager (QMM) 32 is provided to manage the memory 30 in order to and establish and maintain logical queue structures. The QMM 32 also communicates with the control memory 22, as does the update logic block 28. The communication preferably occurs by way of a memory bus (not shown).

Fig. 5A is a logical model of a hierarchical shaper scheduler 40 comprising multiple level 2 shaper sub-schedulers 42 which feed a level 1 exhaustive priority sub-scheduler 44. For an example of how such a hierarchical scheduler may be used and configured, refer to applicants' co-pending patent application U.S.S.N. 09/140,059.

Fig. 5B shows the type of information stored in control memory 22 in relation to scheduler 40. The information stored for a given queue *i* comprises a record 46, termed hereinafter as "$Q_S(i)$", which includes the following fields:

- QUEUE-ID - A unique queue identifier or index.
- E/F - A Boolean value indicating whether the queue is empty or not.
- QD-COUNTER - The queue depth counter.
- TET - The TET for the queue. (This corresponds to TET 15 of Fig. 2.)
- INC - An increment used in time-stamping calculations which corresponds to the shaping rate of the queue.
- SHAPER-ID - An identifier or index of the particular shaper sub-scheduler 42 servicing the queue. Multiple queues may be associated with the same shaper sub-scheduler.

The information stored for a given level 2 shaper sub-scheduler *j* comprises a record 48, termed herein as "$S_S(j)$", which includes the following fields:

- SHAPER-ID - The unique identifier or index of shaper sub-scheduler.
- EX-PRIORITY - A priority level of the shaper sub-scheduler 42 for the purposes of the exhaustive sub-scheduler 44.

Referring to Figs. 4, 5A and 5B, the data processing provided embodied by th control logic block 24 and update logic block 28 in connection with the hierarchical shaper scheduler 40 is discussed in greater detail. In the preferred embodiment, the update logic 28 calculates the TET field in record 46, Qs(*i*) TET, whenever a new packet reaches the head-of-line (HOL) position in the queue. This happens either when

(a) a data packet arrives at an empty queue, or

(b) a data packet has just been served and its queue has a following data packet waiting to be serviced which progresses to the HOL position.

The preferred embodiment employs a virtual clock shaping technique similar to that described in Stiliadios, *supra*. Hence, for arrivals to an empty queue *i*, the update logic 28 calculates the TET for queue i in a conventional manner as follows:

$$Q_S(i).TET=\mathit{max}(RTP, Q_S(i).TET)+ Q_S(i).INC \tag{1}$$

where RTP is a real time pointer or current time maintained by block 24.

In the event a packet is dequeued from queue *i* such that another packet waiting in the queue reaches the HOL position, i.e., if $Q_S(i)$.QD-COUNT > 1, update logic 28 increments the TET of queue *i* by its INC value. That is,

$$Q_S(i).TET= Q_S(i).TET + Q_S(i).INC \tag{2}$$

The control logic 24 builds the tag 17 for each queue which is evaluated by the minimum comparator 26. More specifically, the eligibility field 36 of tag 17 (Fig. 3) is based on:

(a) whether queue *i* is eligible for shaping, i.e., if $Q_S(i).TET - Q_S(i).INC \alpha RTP$; and

(b) whether the queue is non-empty, i.e., if $Q_S(i)$.E/F = FALSE.

If both of these conditions are satisfied then queue *i* is eligible for service and the ELN field 36 is set to Ø, otherwise it is set to 1. These assignments enable the minimum comparator 26 to rank queues which satisfy these conditions ahead of queues which do not.

The control logic 24 sets the level 2 TET field 38 for a given queue *i* to be equal to Q_S(i).TET. The level 1 TET field 37 is set to the priority level of the shaper sub-scheduler 42 serving queue *i*, S_S(Q_S(i).SHAPER-ID).EX-PRIORITY. The tag 17 enables the minimum comparator 26 to rapidly determine which queues are ineligible for service and which queues have the highest priorities, and then to select a winning queue based on the smallest TET value of the queues having the highest priorities.

Fig. 6A is a logical model of a hierarchical WFQ scheduler 50 comprising multiple level 2 WFQ sub-schedulers 52 which feed a level 1 WFQ sub-scheduler 54. For an example of how such an hierarchical scheduler may be used and configured, refer to applicants' co-pending patent application U.S.S.N. 09/140,059.

Fig. 6B shows the type of information stored in control memory 22 in relation to scheduler 50. The information stored for a given queue *i* comprises a record 56, termed herein as "Q_W(i)", which includes the following fields:

- QUEUE-ID - A unique queue identifier or index.
- E/F - A Boolean value indicating whether the queue is empty or not.
- QD-COUNTER - The queue depth counter.
- TET - The TET for the queue. (This corresponds to TET 15 of Fig. 2.)
- INC - An increment used in time-stamping calculations which corresponds to the weight of the queue; as known in the art *per se*.
- WFQ-ID - An identifier of the particular WFQ sub-scheduler 52 servicing the queue. Multiple queues may be associated with the same sub-scheduler 52.

The information stored for a given level 2 WFQ sub-scheduler *j* comprises a record 58, termed herein as "$S_W(j)$", which includes the following fields:

- WFQ-ID - The unique identifier or index of the WFQ sub-scheduler. This field is linked to the WFQ-ID field of the Q_W record 56.
- TET - A time-stamp associated with sub scheduler *j*. This enables the level 1 WFQ sub-scheduler 54 to service the level 2 WFQ sub-schedulers 52 using a weighted fair queuing service discipline.
- INC - An increment used in time-stamping calculations which corresponds to the weight of the level 2 WFQ sub-scheduler 52, as known in the art *per se*.
- RANK - A calculated rank level for the level 2 WFQ sub-scheduler 52.

Referring to Figs. 4, 6A and 6B, the data processing provided by the control logic block 24 and update logic block 28 in connection with the hierarchical WFQ scheduler 50 is discussed in greater detail. In the preferred embodiment, the update logic 28 calculates TETs for each queue 14, and for groups of queues, i.e., for each level 2 WFQ sub-scheduler 52. The TETs for the queues are preferably calculated using the Start Time Fair Queuing (SFQ) model presented in Goyal et al., *supra*. Hence, for arrivals to an empty queue *i*, the update logic 28 calculates the TET for queue *i* as follows:

$$Q_W(i).TET=\mathit{max}(VTP, Q_W(i).TET) + Q_W(i).INC, \qquad (3)$$

where VTP is a virtual time pointer associated with the particular level 2 WFQ sub-scheduler 52 servicing the queue.

In the event a packet is dequeued from queue *i* such that another packet waiting in the queue reaches the HOL position, i.e., if $Q_W(i)$.QD-COUNT > 1, update logic 28 increments the TET of queue *i* by its INC value. That is,

$$Q_W(i).TET = Q_W(i).TET + Q_W(i).INC. \qquad (4)$$

The TET calculations for the level 2 WFQ sub-schedulers 52, however, depart from conventional practice. More specifically, it is typical in WFQ time-stamping techniques to assign TETs only to non-empty queues (or in this case groups of queues). Thus, under conventional practice it is typically required to know whether or not any of the potentially numerous queues serviced by a particular level 2 WFQ sub-scheduler 52 are non-empty in order to know whether the particular level 2 WFQ sub-scheduler 52 can itself be considered to be "empty" and hence ineligible for servicing by the level 1 WFQ sub-scheduler 54. One potential way of accomplishing this task is to introduce a separate signal input representing the empty/full status of each queue associated with a level 2 WFQ sub-scheduler 52 and to AND all such signals together. This approach, however, is inconvenient to implement in practice because a large packet switching device such as a network node could potentially have thousands of queues associated per level 2 sub-scheduler. Also, this approach may impede the programmability of the system in terms of making it more difficult to dynamically specify which queues are serviced by which level 2 sub-scheduler 52.

In order to avoid these limitations, the preferred embodiment employs a novel approach whereby a TET is assigned to each level 2 WFQ sub-scheduler 52 regardless of whether or not its respective queues have data packets waiting for service. More specifically, the update logic 28:

1) Identifies the winning level 2 WFQ sub-scheduler k that was just served based on the output 27 of the minimum comparator 26. In other words, $k = Q_W(x).WFQ\text{-}ID$, where x is the winning queue identifier provided by output 27.

2) Assigns VTP (associated with the level 1 sub-scheduler 54) to be the value of the TET of the winning level 2 WFQ sub-scheduler *k*. In other words, VTP = $S_W(k)$.TET.

3) Updates the TET value of the winning level 2 WFQ sub-scheduler *k* by its pre-determined increment value INC. In other words, $S_W(k)$.TET = $S_W(k)$.TET + $S_W(k)$.INC.

4) Updates the TETs of the remaining level 2 WFQ sub-schedulers 52 by setting their TETs to the maximum of their previous value and VTP. In other words, $S_W(j)$.TET:=*max*($S_W(j)$.TET, VTP), for all $j \Theta k$, (where VTP has been set to the TET of the winning level 2 WFQ sub-scheduler *k* as per step 1).

5) Sorts the level 2 WFQ sub-schedulers 52 in ascending order of their TETs, i.e., in order of increasing $S_W(j)$.TET for all j. The sorted order provides the value for the $S_W(j)$.RANK field, with RANK = Ø for the level 2 WFQ sub-scheduler with the smallest TET, RANK = 1 for the level 2 sub-scheduler with the next largest TET, and so on through the sorted order. In the event of a tie between the TET values of level 2 WFQ sub-schedulers 52, the RANK field may be set in order of their identifiers, i.e., $S_W(j)$.QUEUE-ID, in order to provide a deterministic outcome.

The control logic 24 builds the tag 17 for each queue 14 which is evaluated by the minimum comparator 26. More specifically, the eligibility field 36 of tag 17 (Fig. 3) is based on whether the queue is non-empty, i.e., if Qw(i).E/F = FALSE. If this condition is satisfied then queue *i* is eligible for service and the ELN field 36 is set to Ø, otherwise it is set to 1.

The control logic 24 also sets the level 2 TET field 38 for each queue to be equal to $Q_w(i)$.TET. The level 1 TET field 37 is set to the rank, $S_w(Q_w(i)$.WFQ-ID).RANK, of the level 2 WFQ sub-scheduler 52 serving queue *i*. The tag 17 thus enables the minimum comparator 26 to rapidly determine which queues and level 2 sub-schedulers are ineligible and which of the remaining queues have the highest priorities.

Fig. 7 shows how the foregoing procedure may achieve the principles of SFQ using an example of hierarchical scheduler 50 having five level 2 WFQ sub-schedulers 52. Fig. 7(a) shows the TETs (the nomenclature TET_Z meaning the TET for WFQ sub-scheduler #z) just before service at a given scheduling slot. WFQ sub-scheduler #4 has the smallest TET and WFQ sub-scheduler #1 has the next smallest. Now suppose that WFQ sub-scheduler #4, which has the lowest TET value, has no queue to serve but that WFQ sub-scheduler #1, which has the next lowest TET value, does. The system will thus service a queue associated with WFQ sub-scheduler #1. In accordance with the above procedure, and as shown in Fig. 7(b), the update logic 28 then sets VTP to t_4, the value of TET_1, and increments TET_1 according to its INC value, for example, to t_{10}. Furthermore, as shown in Fig. 7(c), the update logic 28 forces all TETs to at least equal the newly modified VTP such that TET_4 now becomes equal to t_4. This enables the system to conform to the principles of SFQ, whereby VTP must be less than or equal to all TETs. It will also be appreciated this procedure for updating any such "straggling" TETs to equal the value of VTP preserves the relative priority of the level 2 sub-schedulers 52. For instance, if by the time the next scheduling slot arrives WFQ sub-scheduler #4 has a queue to serve, then sub-scheduler #4 will be declared the winner since it will have the lowest TET value (barring a tie, in which case the RANK field may be used as explained previously).

The preferred embodiment has included a sorting step (step 5, above) for providing a priority rank amongst the level 2 WFQ sub-schedulers. It will be understood that the sorting step may be omitted since $S_W(j)$.TET may itself provide the necessary data for the level 1 TET field 37 of tag 17. The step may, however, be desired because it provides a convenient tie-breaking mechanism and also provides a field length which may conveniently be as long as $S_S(j)$.EX-PRIORITY (which is relatively short) such that the same tag structure may be used to implement both WFQ and shaper hierarchical schedulers.

Those skilled in the art will appreciate that many modifications and variations may be made to the preferred embodiment whilst keeping within the spirit of the invention.

CLAIMS

1. An n-level hierarchical scheduler for servicing a plurality of queues, said scheduler comprising:

control logic for assembling a tag associated with each queue, said tag having (i) a field indicative of the eligibility for servicing of the queue, and (ii) n fields, one for each level of the hierarchical scheduler, each field being indicative of the priority of the queue with respect to a corresponding level of the hierarchy;

a comparator for evaluating said tags and providing an output specifying a winning queue based on said evaluation; and

update logic for updating the priority fields of said tag in response to the selection of a winning queue.

2. The scheduler according to claim 1, wherein said fields are concatenated such that the eligibility field assumes a more significant position than the n priority fields in said tag, and the significance of the n priority fields corresponds to their respective level in the hierarchy.

3. The scheduler according to claim 2, wherein the least significant priority field is set to a theoretical emission time (TET) for the corresponding queue.

4. The scheduler according to claim 3, wherein said TET is computed by said update logic in relation to one of a work conserving and a non-work conserving scheduling scheme.

5. The scheduler according to claim 4, wherein said control logic sets the eligibility field associated with each queue based on the eligibility for service of the queue with respect to the selected scheduling scheme.

6. The scheduler according to claim 5, wherein n = 2.

7. The scheduler according to claim 1, wherein said comparator is a minimum comparator.

8. The scheduler according to claim 4, wherein:

the scheduling scheme is a weighted fair queuing (WFQ) work-conserving scheme and the priority fields associated with a given non-bottom level of the hierarchy store theoretical emission times (TET) associated with pre-select groups of queues, and

the update logic (a) sets a virtual time pointer (VTP) equal to the TET of the winning queue, (b) increments TETs associated with the winning queue by pre-selected increment values, and (c) advances any other TET not associated with a bottom level of the hierarchy to equal VTP in the event such TET falls behind the value of VTP.

9. An n-level weighted fair queuing hierarchical scheduler for servicing a plurality of queues, said scheduler comprising:

sub-scheduler processing means for effecting a hierarchy of logical sub-schedulers, including means for logically associating groups of queues with each said logical sub-scheduler at each level of said hierarchy, wherein said processing means produces a theoretical emission time (TET) for each queue or non-top level sub-scheduler;

means for assembling a tag associated with each queue, said tag having (i) a field indicative of the non-empty status the queue and (ii) n fields, one for each level of the scheduling hierarchy, for storing TETs associated with the queue at each level of said hierarchy, wherein said status field assumes a more significant position than said n

TET fields and the significance of said n TET fields corresponds to their respective level in said hierarchy; and

a comparator for evaluating said tags and providing an output specifying a winning queue based on said evaluation,

wherein said processing means (a) sets a virtual time pointer (VTP) equal to the TET of said winning queue, (b) increments said TETs associated with said winning queue by pre-selected increment values, and (c) advances any other given TET associated with one of said logical sub-schedulers to equal said VTP in the event such TET falls behind the value of VTP.

10. A weighted fair queuing method for servicing a plurality of logical queues, comprising:

(a) associating each logical queue with a theoretical emission time (TET) and a predetermined increment value, each logical queue having a defined TET value at all times irrespective of whether or not the logical queue is non-empty;

(b) associating the logical queues as a group with a virtual time pointer (VTP);

(c) selecting one of the logical queues which is non-empty and has a TET value closest to the VTP and thereafter

  (i) retrieving a quantum of data from the selected logical queue and forwarding such data to processing circuitry,

  (ii) setting a new value for VTP equal to the TET of the selected logical queue,

  (iii) incrementing the TET of the selected logical queue by its corresponding increment value, and

(iv) advancing the TET of each remaining logical queue to equal the new value of VTP in the event such TET falls behind the new value of VTP; and

(d) repeating step (c).

11. A weighted fair queue scheduler for selecting amongst a plurality of logical queues contending for bandwidth from a device which processes data packets stored in the queues, the scheduler comprising:

means for associating each logical queue with a theoretical emission time (TET) and a predetermined increment value, each logical queue having a defined TET value at all times irrespective of whether or not the logical queue is non-empty;

means for associating the logical queues as a group with a virtual time pointer (VTP);

means for periodically selecting one of logical queues which is non-empty and has a TET value closest to the VTP, and thereafter

(i) retrieving a quantum of data from the selected logical queue and forwarding such data to the processing device,

(ii) setting a new value for VTP equal to the TET of the selected logical queue,

(iii) incrementing the TET of the selected logical queue by its corresponding increment value, and

(iv) advancing the TET of each remaining logical queue to equal the new value of VTP in the event such TET falls behind the new value of VTP.

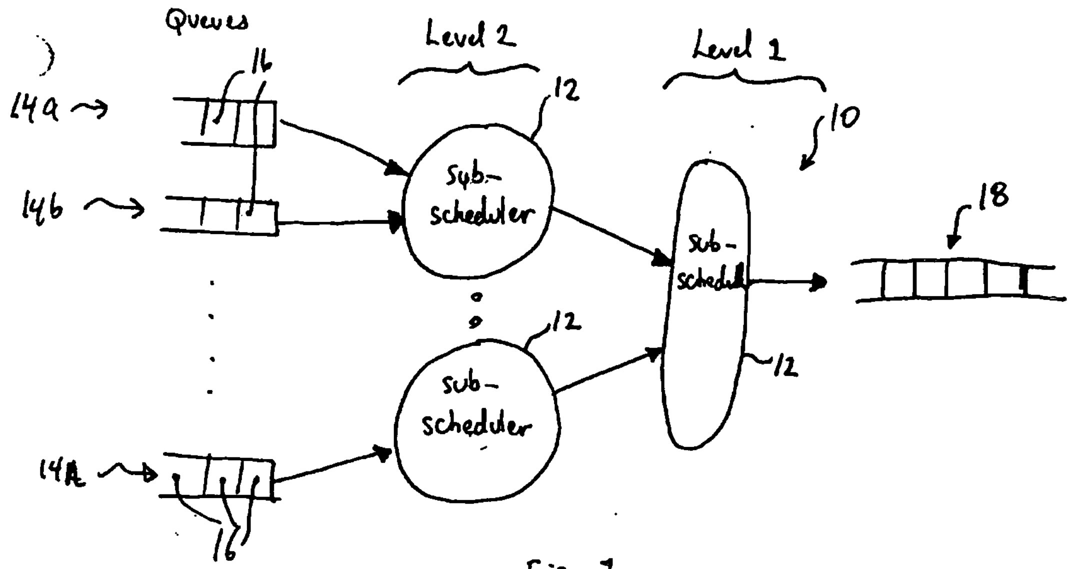


Fig. 1

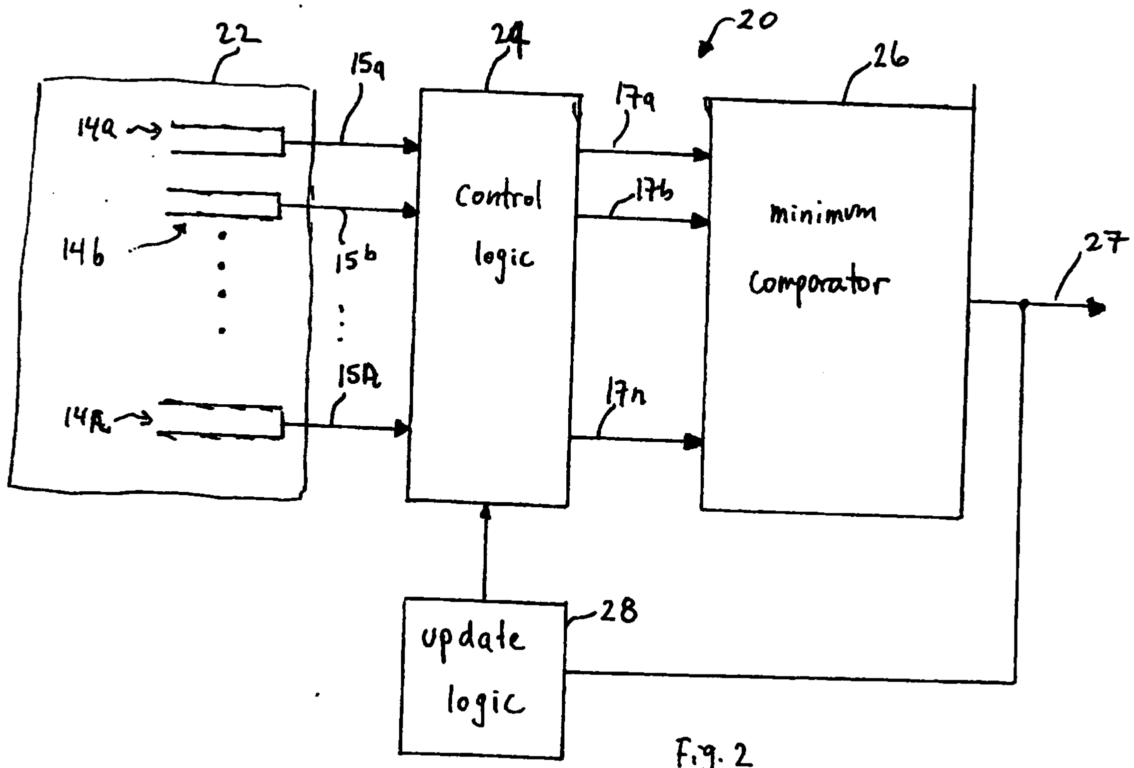


Fig. 2

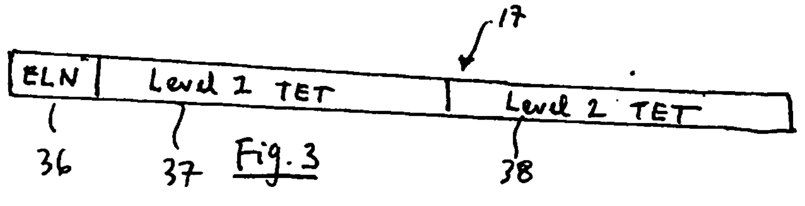


Fig. 3

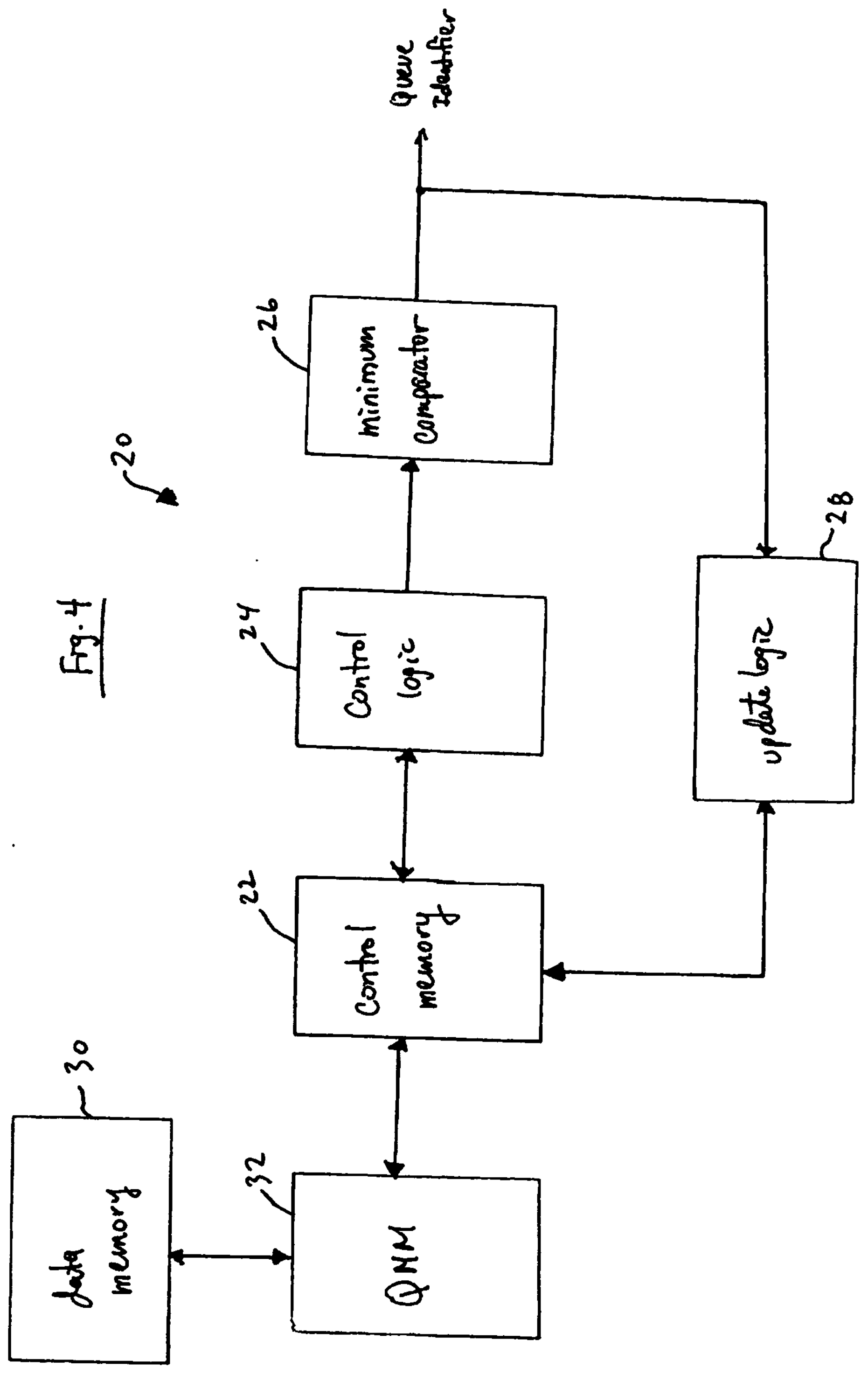

Queue Identifier
26
minimum comparator
20
24
Control logic
28
update logic
22
Control memory
30
data memory
32
QMM

Fig. 4

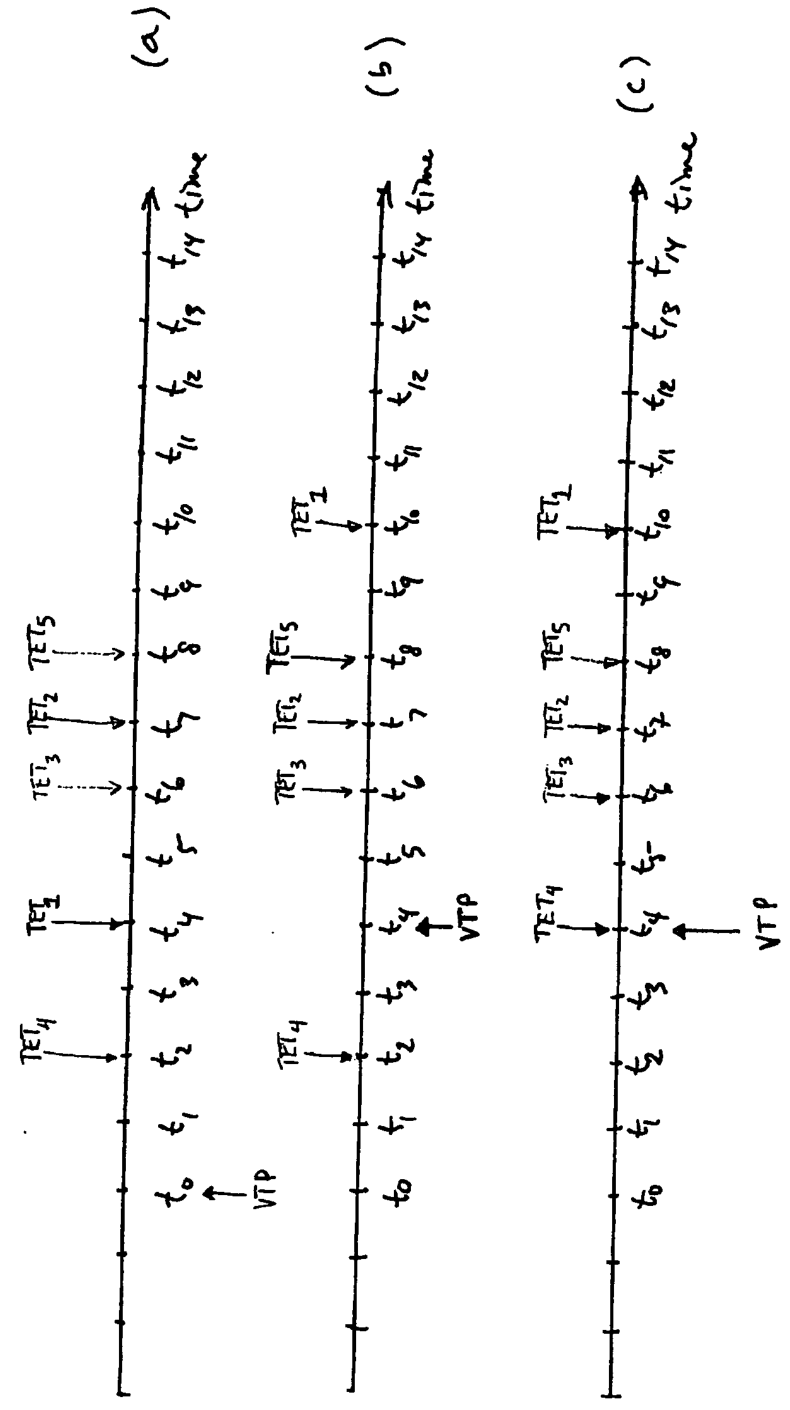
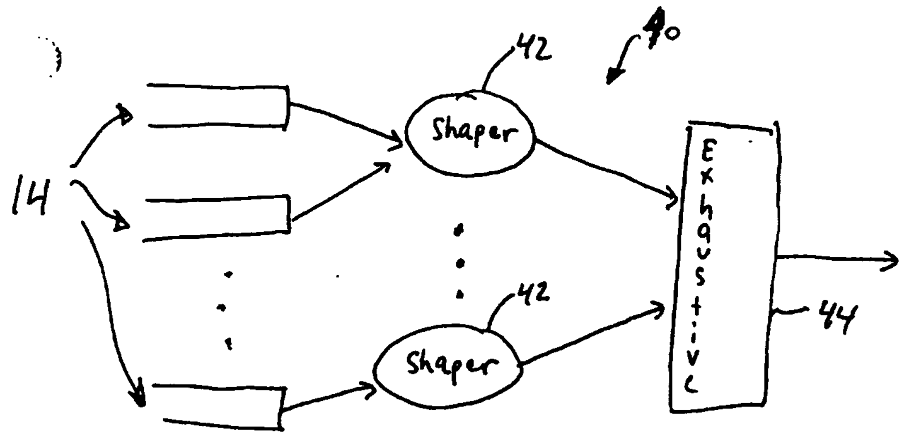
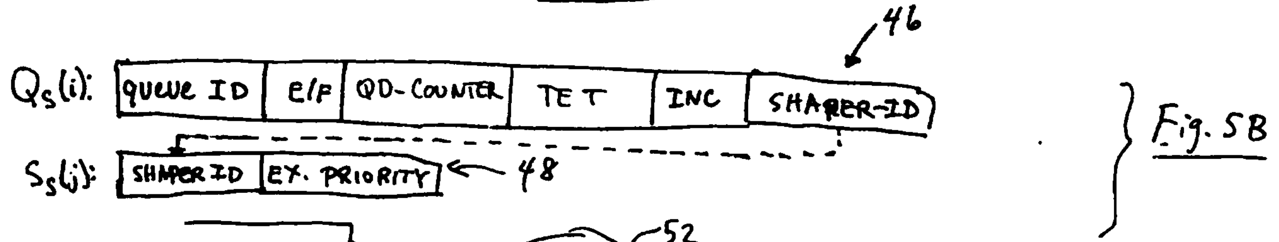

(a)
(b)
(c)
time
t_0
t_1
t_2
t_3
t_4
t_5
t_6
t_7
t_8
t_9
t_{10}
t_{11}
t_{12}
t_{13}
t_{14}
TET_1
TET_2
TET_3
TET_4
TET_5
VTP

Fig. 7

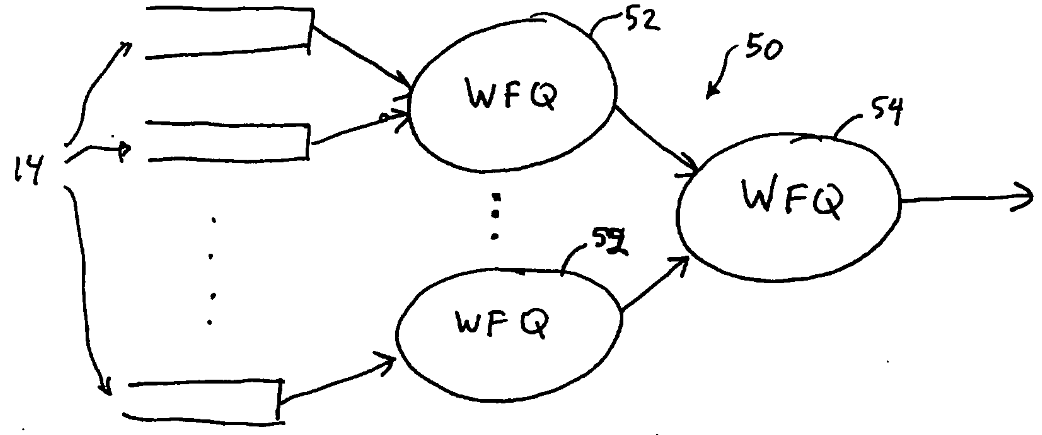


42
40
14
Shaper
Exhaustive
42
Shaper
44

Fig. 5A

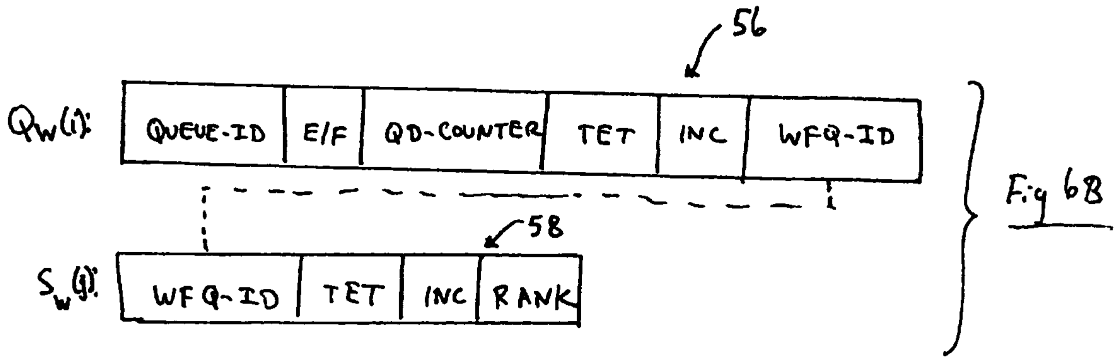


46
Qs(i):
QUEUE ID
E/F
QD-COUNTER
TET
INC
SHAPER-ID
Ss(j):
SHAPER ID
EX. PRIORITY
48
Fig. 5B



52
50
14
WFQ
54
WFQ
52
WFQ

Fig. 6A



56
Qw(i):
QUEUE-ID
E/F
QD-COUNTER
TET
INC
WFQ-ID
58
Sw(j):
WFQ-ID
TET
INC
RANK
Fig 6B